The Routing Performance of Logarithmic-hop Structured P2P Overlay

Saiful Khan
Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia.
saifulkhan@siswa.um.edu.my

Abdullah Gani
Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia.
abdullah@um.edu.my

Mahesh Sreekandath
Qualcomm India Pvt. Ltd., Hyderabad,
mahesh_sree@hotmail.com

Abstract— **This paper presents an analysis on performance of logarithmic degree structured P2P (peer-to-peer) overlay networks. P2P network consist of highly transient peers, where peers join and leave the network randomly also known as dynamic environment. It is, therefore difficult to measure the performance of the parameters in real environment. The design of structured overlay networks is fragmented and due to various designs few simulations have been conducted to compare the protocols in dynamic environment. The outcome of the analysis helps the decision in choosing and designing better structured overlay protocol for P2P network. In order to evaluate the routing performance, this work simulates logarithmic-hop overlays - Chord, Pastry and Kademlia. The result shows that among Chord, Pastry and Kademlia protocols, performance of Kademlia is better than Chord and Pastry with 94.2 - 99% routing efficiency. Hence, Kademlia architecture is better choice to implement structured P2P network.**

Keywords- Chord, Pastry, Kademlia, DHT and P2P

I. INTRODUCTION

The popularity of P2P system is increasing rapidly. There is research focused to design better structured overlay algorithms for P2P systems, especially Distributed Hash Tables (DHT). In DHT based systems identifier (Id) for each peer and item is generated using hashing technique. DHT maps large set of identifiers into set of peers of the structured overlay in distributed fashion; this function is called routing or lookup. The structured overlay protocols Chord, Pastry, Kademlia etc. use DHT. Chord uses consistent hashing and hashes each peer and key into m-bit circular identifier (Id) space, where m is a system parameter [1-2]. During lookup Chord peer uses routing table to route the lookup message to the destination peer. The routing is processed in $O(\log N)$ complexity, where N is total number of peers. In P2P network the routing table of the peers becomes incorrect due to random peers arrival, departure and failure in the network. In Chord peers use periodic stabilization mechanism to correct the routing table periodically. Pastry organizes the peers in 128-bit circular Id space [3]. Pastry uses prefix matching to route the messages and ensures that routing of a message is resolved in $O(\log_b N)$ complexity, where N is total number peers and b is a system parameter. Routing table of Pastry peer is maintained periodically by exchanging keep-alive messages between the neighboring peers. Kademlia assigns 160-bit Id to each peer and each Kademlia peer contains list of entries called routing buckets [4]. Kademlia uses XOR based closeness algorithm to calculate the closest peers and sends parallel lookup messages to multiple buckets. Kademlia routes the messages at complexity of $O(\log_b N)$, where N is total number of peers and b is system parameter. Kademlia peer employs maintenance algorithm which periodically checks the routing buckets through which there has not been a lookup since last stabilization. In order to do so Kademlia performs lookup for each bucket's binary prefix. Kademlia only ensures that at least one entry in each bucket is alive since last stabilization, where as Chord and Pastry ensures all routing entries are alive and up to date.

Structured overlays use its corresponding routing algorithm and the routing table to route the messages to the destination peer. The P2P system consists of peers, where peers join and leaves the network continuously. The random join and leave of peers causes the routing table incorrect and out of date. As a result, the routing timeout and eventually lookup failure occurs. The rate of failure is dependent on the rate of peer join and leave or mean life time of the peers. On the other hand each structured overlay uses the maintenance algorithm to keep the routing table up to date. The correctness of the routing table depends on how frequently the routing table is maintained. Therefore, the routing performance increases with the increase in frequency of maintenance. The asymptotic analysis of routing performance of the structured overlay protocols - Chord, Pastry and Kademlia are logarithmic. The main challenge is to evaluate the performance in a real environment where peers join and leave randomly. A good analysis will enable to decide in choosing and designing better structured overlay for P2P network. In this paper the simulation of Chord, Pastry and Kademlia is presented in order to evaluate the routing performance in P2P network.

II. ARCHITECTURE OF THE PROTOCOLS

A. Chord

Chord is the first structured P2P overlay protocol which is based on DHT. Chord protocol uses consistent hashing technique to map each peer and item (key) to an m-bit identifier circle [2]. The Ids are drawn from the range $[0 - 2^{m-1})$. As shown in Figure 1, the successor of a key and peer is the immediate next peer in the identifier circle towards clockwise direction. The predecessor of a key and peer is the immediate next peer in the identifier circle towards anti-clockwise direction.

Routing table of Chord peer is divided into two parts. The first part consists of its predecessor peer and the second part consists of successor peers. The i^{th} entry of routing table of a peer n contains the address of the peer whose Id is closest to $n + 2^{i-1}$ in clock-wise direction. Thus the first entry of routing table of any peer is the address of the successor of that peer. When a peer issues a key lookup, the lookup message is routed through the identifier circle. The peer first checks if the value of the key Id is between its own peer Id and its successor's peer Id. If the value of the key Id falls between, the successor is returned as the destination peer. Otherwise the lookup message is routed to the numerically closest peer of its routing table. The routing of the lookup message proceeds iteratively or recursively until the destination peer is discovered. Chord takes $O(\log N)$ number of steps to route a message to destination peer, where N is the total number peers. Peer failure or departure affects the routing correctness. The departure of a peer causes its predecessor peer's successor pointer invalid, so this problem is addressed by maintaining a successor list. The successor list is stabilized using periodic stabilization mechanism. Figure 1 shows an example of routing scenario in Chord, where a peer 8 issues a lookup for key-54. Peer 8 invokes the find successor method for this key, this returns the address of the successor of key-4, and here it is peer 56. Newly joined Chord peer first generates peer Id using consistent hashing technique. Then the new peer contacts a bootstrapping peer (an existing peer in Chord network) and issues a lookup. The result of the lookup is the Id of its successor peer. Peer uses periodic stabilization mechanism to correct its routing table completely. Stabilization protocol has two major functions described below:

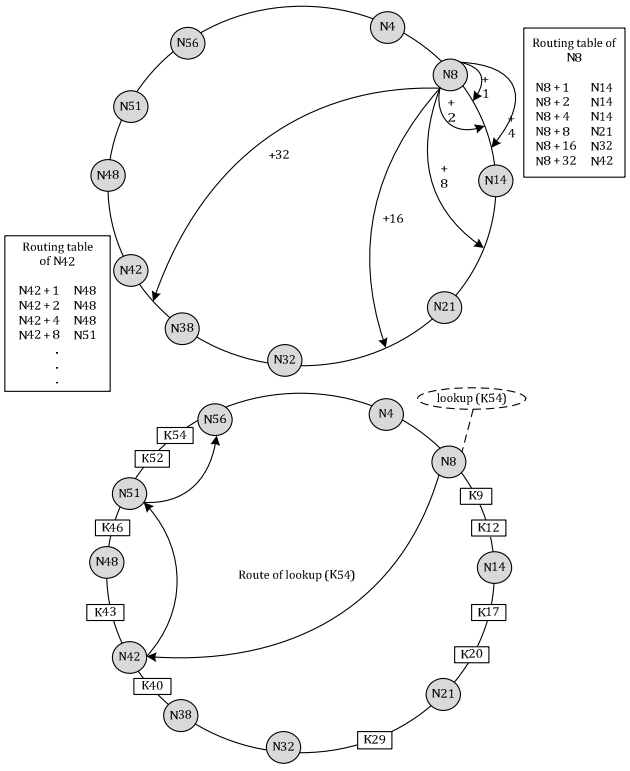

Figure 1. Example routing scenario in Chord [5].

- stabilize( ): This function is periodically executed by each peer to learn about newly joined peers in the chord network and update the routing table accordingly.
- fix_fingers(): This function is periodically executed by each peer to correct the polluted routing table (finger table) caused due to peer departure from the network.

B. Pastry

In Pastry routing table of each peer is organized in proximity metric which uses prefix based matching algorithm to route the messages. Pastry organizes the peers into a 128-bit circular Id space. Peer Id is generated by hashing the IP address of the peer. Each peer has leaf set containing $\frac{L}{2}$ successor and $\frac{L}{2}$ predecessors. A peer also has neighborhood set which keeps track of M peers that are close according to metric other than the Id space (e.g. network delay). Neighborhood set is used for maintaining locality properties. Routing table of Pastry peer is organized into two dimensions with $\log_b N$ rows and $2^b - 1$ entries per row. The $2^b - 1$ entries at row n of the routing table refer to a peer whose peer Id shares the present peer's Id in the first n digits, but whose $(n+1)^{th}$ digit has one of the $2^b - 1$ possible values other than the $(n+1)^{th}$ digit in the present peer's Id.

Upon receiving a key lookup a peer first checks the numerically closest peer of the key Id. Then the peer routes the lookup to the numerically closest peer within $\log_{2^b} N$ hops. The routing process has two main steps. At first the peer received the key lookup checks whether the key is within the range of its leaf set. If the key is within the range of its leaf set the message is sent directly to the destination peer. Otherwise, in second step the message is forwarded to the peer that shares a common prefix with the key by at least one more digit. An example lookup scenario of Pastry network is presented in Figure 2. The peer 859fdc looks for a key d57b2d . From routing table, the peer gets d13a14 , where the peer d13a14 and key d57b2d shares one digit common prefix. Then peer d13a14 checks its routing table and gets d52acd, where the peer d52acd and key d57b2d shares one more digit as common prefix. The routing of the lookup message proceeds until the key is discovered by the peer d57b0c. When a new peer joins a Pastry network it contacts a bootstrapping peer in the network and issues a join message with its peer Id. This message is routed to the peer which is numerically closest to new peer Id. All the peers encountered on the routing path send their routing tables to the new peer. The peer then initializes its routing table based on the received information. Routing table of Pastry peer is maintained periodically by exchanging keep-alive messages among neighbor peers. Upon detecting peer failure, all members of the failed peer's leaf set are notified and they update their corresponding leaf sets.

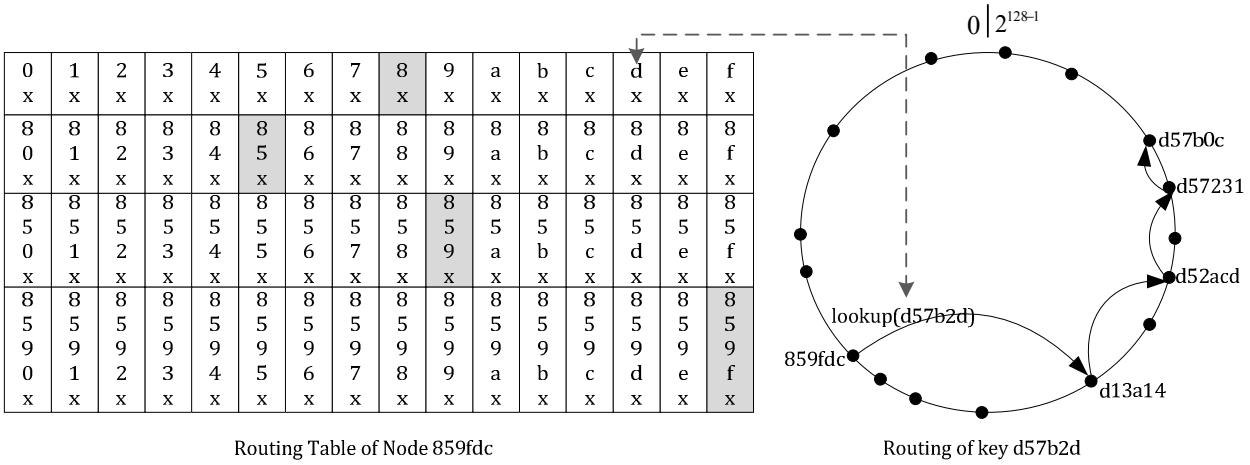

Figure 2. Example Pastry routing scenario [6].

C. Kademlia

Kademlia assigns 10-bit peer Id and calculates the closeness between the peers by taking XOR of the peer Ids. In XOR based closeness algorithm, the distance between two peers would be higher if their Ids are different at higher order bits. The routing table of Kademlia also called routing bucket, which consist of $\log N$ buckets and each bucket consist of K entries. Each peer is placed in a bucket, where bucket i contains the peers at distance $[2^i - 2^{i+1})$. Kademlia employs two new mechanisms in overlay routing: Kademlia sends parallel lookup requests and exchanges routing table information during lookup, instead of using separate maintenance messages.

Kademlia sends lookup message concurrently to multiple buckets and the lookup is routed in iterative manner. During lookup a peer computes the XOR distance to the destination peer. During the same process, it checks the k-buckets from its routing table to choose $\propto$ closest peers randomly and sends parallel request to those peers. Each lookup response returns closer peers, because routing tables are organized to have higher density in the vicinity of the peer. Each peer returns back k-bucket of a smaller sub-tree which is closer to the Id. From the returned buckets another $\propto$ randomly selected peers are contacted. This process is repeated until the Id is found. When an item is inserted, this is also stored at the k closet peers to its Id. The routing of a message in Kademlia is resolved in $O(\log N)$ hops.

When a new peer joins the Kademlia network it performs lookup for its peer Id. In that lookup process, the peers along the query path get to know about the newly joined peer. Similarly the departure of a peer is discovered only when some other peers attempt to ascertain the departed peer's presence before evicting it from a bucket. Unlike periodic stabilization of Chord and the deterministic update of Pastry, Kademlia does not employ any separate maintenance algorithm to keep the routing buckets up to date in case of peer join and leave. Kademlia makes use of the lookup traffic to maintain the routing tables. The reception of any message from a certain peer in a certain sub-tree is essentially an update of the k-bucket for that sub-tree. This approach minimizes the maintenance messages and same in turn minimizes the maintenance bandwidth. Figure 3 shows an example Kademlia identifier space of size $N = 16$ and a peer with Id 3 (00112). Since its prefix of length 1 is the digit 0, it needs to know a peer whose first digit is 1. Since its prefix of length 2 is 00, it needs to know a peer with prefix 01. Similarly, since its prefix of length 3 is 001, it needs to know a peer with prefix 000. Finally, as its prefix of length four is 001, it needs to know a peer with a prefix 0010. This policy is illustrated in Figure 3 which results in a space division exactly like Pastry with the special case of a binary encoding of the digits.

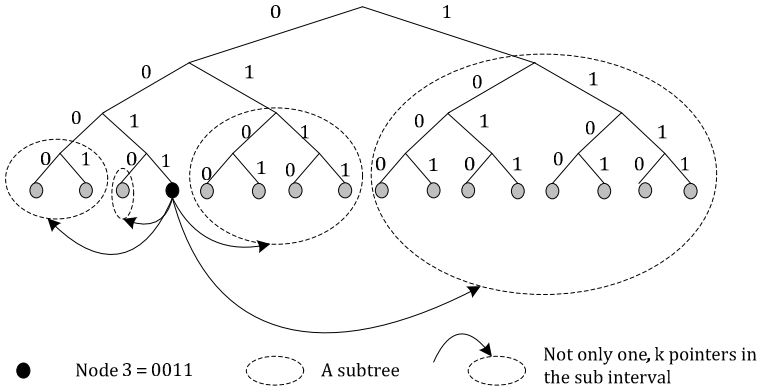

Figure 3. Pointers of peer 3 in Kademlia overlay structure.

III. RELATED WORK

S. Rhea, et al [6] simulated FreePastry and found that, most of the messages fail to route when the life-time of the peers are less. Short-lived peers leave the overlay network with lookups that has not yet timed out. Reactive recovery increases the traffic to congested links, so the usage of periodic recovery has been suggested. It has been suggested that a good lookup timeout value should be calculated by taking exponential moving average (EMA) of each neighbor's response time, instead of considering a fixed timeout. The nearby neighbor should be selected by global sampling, instead of sampling the neighbour's neighbor. M. Castro, et al. [7] used the implementation of MSPastry and showed that it can perform well in P2P network by achieving shorter routing paths and a lesser maintenance overhead. Liben-Nowell et al. [8] shown that limited amount of housekeeping work improve the routing efficiently. Stoica et al. [1] suggested that in Chord recursive routing algorithm perform better than the iterative routing algorithm. L. Alima, et al [9] proposed correction-on-use mechanism in Distributed K-ary Search (DKS), which is similar to Chord protocol, but reduce the maintenance overhead incurred by Chord stabilization. L. Liu, et al. [7] presented a small world architecture network (SWAN) for P2P, for resource routing in multi group P2P system. This paper described by analysis and simulation that SWAN model is better choice for P2P network compared to existing structured and unstructered overlay. J. Li, et al [8] proposed PVC framework which is a direct way to compare Structured Overlay protocols: Kademlia, Kelips, Tapestry and OneHop using cost metric. The cost metric calculates the average number of bytes sent per peer per unit time. In PVC Structured Overlay protocols are compared by simulation. Simulation was done using various combinations of parameters. And result shows that the performance of Tapestry protocol can be optimally tuned by tuning the parameters: base, the stabilization frequency, number of backup peers and number of peers contacted during repair. Finding in PVC framework shows that Chord performs better than other Structured overlay protocols, when the bandwidth is low. Chord has low maintenance ring structure, so low bandwidth is sufficient to maintain Chord; whereas low bandwidth is not sufficient for other protocols. When bandwidth increases beyond 25byes/sec Chord's performance increment is narrow and One-hop protocol performs best.

IV. PERFORMANCE EVALUATION

A. Asymptotic Performance

Table I presents the asymptotic routing performance of the protocols. The asymptotic performance of the protocols is analogous and in the order of logarithm. The performance of the protocols in P2P network where peers join and leave the network randomly is evaluated by simulating the protocols.

B. Simulation Framework

The simulation framework is 3-tier architecture, which is setup in OverSim[9] simulator using NED language[10]. The 3-tier simulation framework is presented in Figure 4, which has following layers.

a) Application Layer: OverSim has KBR test application implemented on it. KBR periodically sends the test messages to the random peers and records different parameters such as message delay, hop-count and so on.

b) Overlay Layer: Structured overlay protocols like Chord, Pastry, Kademlia and so on are implemented in OverSim. The communication between Overlay and Application happens through common API.

c) Underlay Layer: The simulation framework use INET underlay as the underlay later. INET includes simulation models of all network layers starting from MAC and onwards.

In simulation the statistics for routing performance for different mean life-time of the peers are calculated and recorded. The value of the important system parameters and the mean life-time of the peers are mentioned in Table II. The evaluation of performance using real implementation is limited to the network which consist of hundred peers [11-17]

TABLE I. ASYMPTOTIC PERFORMANCE

Criteria	Architecture		
	Chord	***Pastry***	***Kademlia***
Routing Performance	$O(\log N)$	$O(\log_b N)$	$O(\log_b N)$

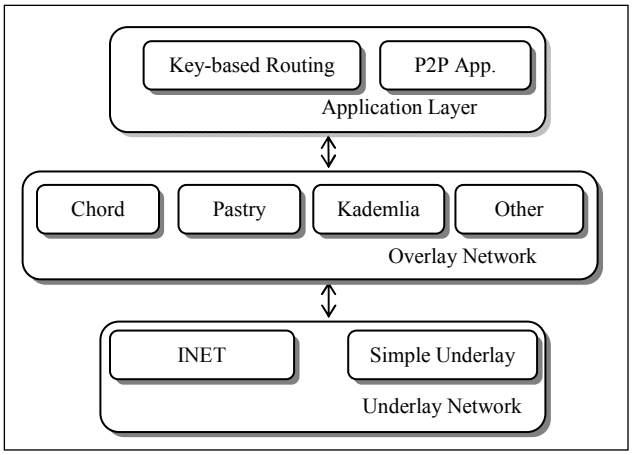

Figure 4. The 3-tier Simulation Framework

TABLE II. PARAMETERS USED FOR SIMULATION

Name of the Parameter	Architecture		
	Chord	***Pastry***	***Kademlia***
Number of peers	10K	10K	10K
Mean life-time of the peer (sec)	500,1000,2000, 5000,7500	500,1000,2000, 5000,7500	500,1000,200 0,5000,7500
Other parameters	No. of successor = 8, Stabilize retry = 1, Stabilize delay = 20 sec	Leaf set size = 16, Neighbor set size = 32, No. of bits per digit = 4	Parallel lookups = 3, Peers per entry = 8, Number of Ids returned = 8

and in simulation the number of peers were limited to 20000 peers [11-13] [1, 16, 18-23]. Hence this simulation the network consist of 10000 overlay peers. The mean life-time of the peers are varied between 500 to 7500 sec, which bear similarities with most of the previously conducted experiments [15] [16] [24-25]. The system parameters are default value [9, 26]. Three sets of simulations are executed, where peers join and leave the network based on times drawn from a Weibull distribution, recommended by [27]. KBR application layer periodically sends the test messages to the random overlay peers (Ids) and statistics for different parameters are recorded in corresponding layers. If the data item searched is found at least once, the search is considered to be a successful lookup.

V. RESULTS AND DISCUSSION

The average values of the parameters recorded in the simulation is plotted with 95% confidence interval. Table III presents the value of routing efficiency for different mean life-time of the peers. Figure 5 illustrate the routing efficiency Vs mean life-time of the peers of Chord, Pastry and Kademlia. The value of routing efficiency is calculated by taking the ratio of number of messages routed successfully and total number of messages sent. In Chord the routing efficiency is 25.4% when the mean life-time of the peers is 500 sec. When the mean life-time of peer is less than 1000 sec most of the messages fails to reach the destination. The routing efficiency increases with the increase in mean life-time of the peers and when mean life-time is greater than 7500 sec it saturates at 95%. In Pastry the routing efficiency is 90.3% when the mean life-time of the peers is 500 sec. The lookup routing efficiency increases with the mean life-time of the peers and saturates to 99% when mean life-time is greater than 7500 sec. Therefore, Pastry performs better than Chord. In Kademlia the routing efficiency is 94.2% when the mean life-time of the peers is 500 sec. Te routing performance of Kademlia increases with the mean life-time of the peers and saturates at 99% when mean life-time is greater than 7500 sec. Chord organizes the routing table in single dimensional space. Pastry and Kademlia has bigger routing table size and organizes the routing table in two-dimensional space. Unlike Chord, Kademlia and Pastry calculates the network locality by using prefix matching and XOR based closeness algorithm.

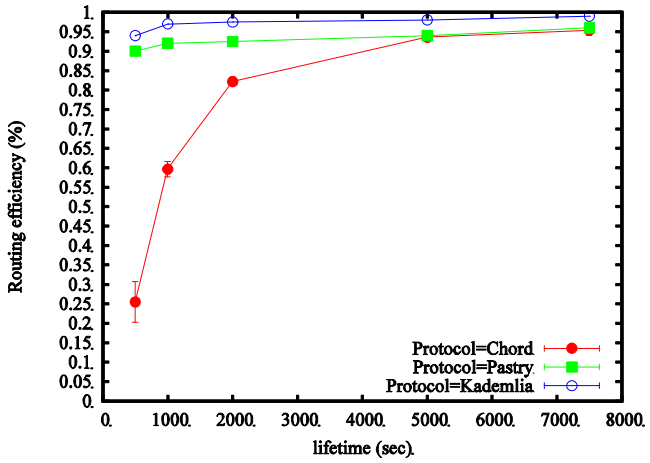

Figure 5. Mean life-time Vs routing efficiency

TABLE III. ROUTING EFFICIENCY

Mean life-time of the peer (sec)	Percentage of success (%)		
	Chord	*Pastry*	*Kademlia*
500	25.4	90.3	94.2
1000	59.6	92.0	97.0
2000	82.1	92.5	97.4
5000	93.6	94.2	98.1
7500	95.4	96.3	99.0

Thus Kademlia and Pastry minimizes the distance a particular lookup needs to travel to discover the destination peer. Hence Pastry and Kademlia search the destination peer faster than Chord.

VI. CONCLUSION

In this paper the simulation of three logarithmic-hop structures overlay protocols - Chord, Pastry and Kademlia is presented. The simulation is conducted in a network which consists of 10,000 peers where peers join and leave the network randomly. The simulation evaluates the routing performance of the mentioned protocols. The result shows that in P2P network, the increase in mean life-time of the peers improves the routing performance. When the mean life-time of the peers is less Chord is more dependent on stabilization frequency. Pastry has better routing performance than Chord. Among Chord, Pastry and Kademlia protocols, Kademlia performs better than Chord and Pastry with 94.2 - 99% routing efficiency. Future research is required in order to evaluate the performance of additional logarithmic-hop as well as single-hop overlays. Future research is also required to evaluate other important parameters such as maintenance bandwidth consumption, searching efficiency, hop count and so on in a network of large number of peers.

REFERENCES

[1] I. Stoica, *et al.*, "Chord: A scalable peer-to-peer lookup service for Internet applications," *Computer Communication Review,* vol. 31, pp. 149-160, Oct 2001.

[2] D. Karger, *et al.*, "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web," presented at the Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, El Paso, Texas, United States, 1997.

[3] P. Druschel and A. Rowstron, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science,* vol. 2218, 2001.

[4] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Peer-to-Peer Systems,* pp. 53-65, 2002.

[5] L. Eng Keong, *et al.*, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials, IEEE,* vol. 7, pp. 72-93, 2005.

[6] K. Dhara, *et al.*, "Overview of Structured Peer-to-Peer Overlay Algorithms," *Handbook of Peer-to-Peer Networking,* pp. 223-256, 2010.

[7] L. Liu, *et al.*, "Fault-tolerant peer-to-peer search on small-world networks," *Future Generation Computer Systems,* vol. 23, pp. 921-931, 2007.

[8] J. Li, *et al.*, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," 2005, pp. 225-236.

[9] I. Baumgart, *et al.*, "OverSim: A Flexible Overlay Network Simulation Framework," in *IEEE Global Internet Symposium, 2007*, 2007, pp. 79-84.
[10] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," presented at the Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, Marseille, France, 2008.
[11] F. Dabek, *et al.*, "Wide-area cooperative storage with CFS," presented at the Proceedings of the eighteenth ACM symposium on Operating systems principles, Banff, Alberta, Canada, 2001.
[12] P. Fonseca, *et al.*, "Full-Information Lookups for Peer-to-Peer Overlays," *Ieee Transactions on Parallel and Distributed Systems,* vol. 20, pp. 1339-1351, 2009.
[13] R. Huebsch, *et al.*, "Querying the internet with PIER," presented at the Proceedings of the 29th international conference on Very large data bases - Volume 29, Berlin, Germany, 2003.
[14] S. Rhea, *et al.*, "Pond: The OceanStore Prototype," presented at the Proceedings of the 2nd USENIX Conference on File and Storage Technologies, San Francisco, CA, 2003.
[15] S. Rhea, *et al.*, "Handling churn in a DHT," presented at the Proceedings of the annual conference on USENIX Annual Technical Conference, Boston, MA, 2004.
[16] B. Zhao, *et al.*, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on selected areas in communications,* vol. 22, pp. 41-53, 2004.
[17] Z. Li and P. Mohapatra, "On investigating overlay service topologies," *Computer Networks,* vol. 51, pp. 54-68, 2007.
[18] J. Y. Li, *et al.*, "Comparing the performance of distributed hash tables under churn," *Peer-to-Peer Systems Iii,* vol. 3279, pp. 87-99, 2004.
[19] J. Y. Li, *et al.*, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," in *IEEE Infocom 2005: The Conference on Computer Communications, Vols 1-4, Proceedings*, K. Makki and E. Knightly, Eds., ed, 2005, pp. 225-236.
[20] A. T. Mizrak, *et al.*, "Structured superpeers: leveraging heterogeneity to provide constant-time lookup," in *Internet Applications. WIAPP 2003. Proceedings. The Third IEEE Workshop on*, 2003, pp. 104-111.
[21] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," *SIGOPS Oper. Syst. Rev.,* vol. 35, pp. 188-201, 2001.
[22] K. Y. K. Hui, *et al.*, "Small-world overlay P2P networks: Construction, management and handling of dynamic flash crowds," *Computer Networks,* vol. 50, pp. 2727-2746, 2006.
[23] X. Liu, *et al.*, "Consistency maintenance in dynamic peer-to-peer overlay networks," *Computer Networks,* vol. 50, pp. 859-876, 2006.
[24] L. Monnerat and C. Amorim, "Peer-to-Peer Single Hop Distributed Hash Tables," *Globecom 2009 - 2009 Ieee Global Telecommunications Conference, Vols 1-8,* pp. 4250-4257, 2009.
[25] Z. Ou, *et al.*, "Performance evaluation of a Kademlia-based communication-oriented P2P system under churn," *Computer Networks,* vol. 54, pp. 689-705, 2010.
[26] J. P. Munoz-Gea, *et al.*, "Simulation of a P2P Application Using OverSim," in *Advances in Future Internet, 2009 First International Conference on*, 2009, pp. 53-60.
[27] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," presented at the Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, Rio de Janeriro, Brazil, 2006.